```
# Demostración vueloEscalerado

## Transformacion de estados

bool Drone::vueloEscalerado() const
{
        bool escalerado = true;
        // Estado 1
        // vale escalerado == true
        // vale Pif: true
        if (vueloRealizado().size() >= 2) {
                // Estado if1
                // vale Pif ∧ escalerado == escalerado@E1
                // vale Bif: (res == vueloRealizado(this)) (por asegura de vueloRealizado) ∧ (|res| >= 2)
                // implica Bif: |vueloRealizado(this)| >= 2 (reemplazo sintáctico)
                int i = 0;
                // Estado if2
                // vale Pc: (i == 0) ∧ (escalerado == escalerado@Eif1) ∧ Bif (porque no se modifico vueloRealizado)
                while (i < vueloRealizado().size() - 2) {
                        // I: (0 <= i <= |vueloRealizado(this)| - 2) ∧ (((∀m←[0..i))(∃x,y←[1,-1])
                        //      ((prm(vueloRealizado(this)[m]) - prm(vueloRealizado(this)[m+2])) == x) ∧
                        //      ((sgd(vueloRealizado(this)[m]) - sgd(vueloRealizado(this)[m+2])) == y)) == escalerado)
                        // V: |vueloRealizado(this)| - 2 - i
                        // C: 0
                        // Bc: i < |vueloRealizado()| - 2
                        // Estado c1
                        // vale I ∧ Bc ∧ (escalerado == escalerado@Eif2)
                        int deltaX = vueloRealizado()[i].x - vueloRealizado() [i+2].x;
                        // Estado c2
                        // vale (i == i@Ec1) ∧ (escalerado == escalerado@Ec1) ∧
                        //      (deltaX == (prm(vueloRealizado(this)[i]) - prm(vueloRealizado(this)[i+2])))
                        int deltaY = vueloRealizado()[i].y - vueloRealizado() [i+2].y;
                        // Estado c3
                        // vale (i == i@Ec2) ∧ (escalerado == escalerado@Ec2) ∧
                        //      (deltaX == (prm(vueloRealizado(this)[i]) - prm(vueloRealizado(this)[i+2]))) ∧
                        //      (deltaY == (sgd(vueloRealizado(this)[i]) - sgd(vueloRealizado(this)[i+2])))
                        // vale Pif1: (deltaX == deltaX@Ec3) ∧ (escalerado == escalerado@Ec3)
                        if ((deltaX != 1) && (deltaX != -1)) escalerado = false;
                        // Estado c4
                        // vale Qif1: (((deltaX ≠ 1) ∧ (deltaX ≠ -1)) → escalerado == false) ∧
                        //      ((deltaX == 1) ∨ (deltaX == -1)) → escalerado == escalerado@Ec3))
                        // vale (i == i@Ec3) ∧ (deltaY == deltaY@Ec3)
                        // vale Pif2: (deltaY == deltaY@Ec3) ∧ (escalerado == escalerado@Ec4)
```

```
                        if ((deltaX != 1) && (deltaX != -1)) escalerado = false;
                        // Estado c5
                        // vale Qif2: (((deltaY ≠ 1) ∧ (deltaY ≠ -1)) → escalerado == false) ∧
                        //      ((deltaY == 1) ∨ (deltaY == -1)) → escalerado == escalerado@Ec3))
                        // vale (i == i@4) ∧ (deltaX == deltaX@4)
                        i++;
                        // Estado c6
                        // vale (i == i@Ec5 + 1) ∧ (deltaX == deltaX@Ec5) ∧ (deltaY == deltaY@Ec5) ∧ (escalerado ==
escalerado@Ec5)
                        }
                // Estado if3
                // vale Qc: (i == (|vueloRealizado(this)| - 2)) ∧ (escalerado == escalerado(vueloRealizado(this))) (usando la
aux escalerado)
        }
        // Estado 2
        // vale Qif: Qc
        return enVuelo() && escalerado;
        // Estado Q
        // vale res == enVuelo(this) (por asegura de enVuelo) ∧ (result == (res ∧ escalerado))
        // implica (result == (enVuelo(this) ∧ escalerado)) ∧ Qif (reemplazo sintáctico)
        // implica (result == (enVuelo(this) ∧ escalerado)) ∧ Qc (reemplazo sintáctico)
        // implica (result == (enVuelo(this) ∧ escalerado)) ∧ (escalerado == escalerado(vueloRealizado(this))) ∧
        //      (i == (|vueloRealizado(this)| - 2)) (reemplazo sintáctico)
        // implica (result == (enVuelo(this) ∧ escalerado(vueloRealizado(this))) ∧
        //      (i == (|vueloRealizado(this)| - 2)) (reemplazo sintáctico)
        // implica (result == (enVuelo(this) ∧ escalerado(vueloRealizado(this)))
}

-----------------------------------------------------------------------------------------------------

## Demostraciones ciclo

- Pc → I
        vale Pc: (i == 0) ∧ (escalerado == escalerado@Eif1) ∧ Bif
        implica (i == 0) ∧ (escalerado == escalerado@E1) ∧ |vueloRealizado(this)| >= 2 (reemplazo sintáctico)
        implica (i == 0) ∧ (escalerado == true) ∧ (|vueloRealizado(this)| >= 2) (reemplazo sintáctico)
        implica (i == 0) ∧ (escalerado == true) ∧ (|vueloRealizado(this)| - 2 >= 0)
        implica (i == 0) (escalerado == true) ∧ (|vueloRealizado(this)| - 2 >= i) (reemplazo sintáctico)
        implica (escalerado == true) ∧ (0 <= i <= |vueloRealizado(this)| - 2)

        implica (((∀m←[0..i))(∃x,y←[1,-1])
                ((prm(vueloRealizado(this)[m]) - prm(vueloRealizado(this)[m+2])) == x) ∧
                ((sgd(vueloRealizado(this)[m]) - sgd(vueloRealizado(this)[m+2])) == y)) == escalerado)
```

```
        implica (true == escalerado) (porque ∀m←[0..i) → ∀m←[0..0) que es siempre true)
        implica (true == true) (reemplazo sintactico)

         implica I: (0 <= i <= |vueloRealizado(this)| - 2) ∧ (((∀m←[0..i))(∃x,y←[1,-1])
                ((prm(vueloRealizado(this)[m]) - prm(vueloRealizado(this)[m+2])) == x) ∧
                ((sgd(vuelorealizado(this)[m]) - sgd(vuelorealizado(this)[m+2])) == y)) == escalerado)
```

```
- (I ∧ ¬Bc) → Qc
         vale I: (0 <= i <= |vueloRealizado(this)| - 2) ∧ (((∀m←[0..i))(∃x,y←[1,-1])
                ((prm(vueloRealizado(this)[m]) - prm(vueloRealizado(this)[m+2])) == x) ∧
                ((sgd(vueloRealizado(this)[m]) - sgd(vueloRealizado(this)[m+2])) == y)) == escalerado)
         vale ¬Bc (i >= |vueloRealizado()| - 2)
         implica (i == |vueloRealizado(this)| - 2) (porque (i <= |vueloRealizado(this)| - 2) ∧ (i >= |vueloRealizado()| - 2))
         implica (i == |vueloRealizado(this)| - 2) ∧ (((∀m←[0..i))(∃x,y←[1,-1])
                ((prm(vueloRealizado(this)[m]) - prm(vueloRealizado(this)[m+2])) == x) ∧
                ((sgd(vueloRealizado(this)[m]) - sgd(vueloRealizado(this)[m+2])) == y)) == escalerado)
         implica (i == |vueloRealizado(this)| - 2) ∧ (((∀m←[0..|vueloRealizado(this)| - 2))(∃x,y←[1,-1])
                ((prm(vueloRealizado(this)[m]) - prm(vueloRealizado(this)[m+2])) == x) ∧
                ((sgd(vueloRealizado(this)[m]) - sgd(vueloRealizado(this)[m+2])) == y)) == escalerado)
                (reemplazo sintactico)
         implica (i == |vueloRealizado(this)| - 2) ∧ escalerado(vueloRealizado(this))

- (I ∧ (V < C)) → ¬Bc
         vale I: (0 <= i <= |vueloRealizado(this)| - 2) ∧ (((∀m←[0..i))(∃x,y←[1,-1])
                ((prm(vueloRealizado(this)[m]) - prm(vueloRealizado(this)[m+2])) == x) ∧
                ((sgd(vueloRealizado(this)[m]) - sgd(vueloRealizado(this)[m+2])) == y)) == escalerado)
         vale (V < C): (|vueloRealizado(this)| - 2 - i) < 0
         implica (|vueloRealizado(this)| - 2) < i
         implica ¬(i <= (|vueloRealizado(this)| - 2))
         implica ¬Bc: ¬(i < (|vueloRealizado(this)| - 2))
```